Claims

1. A program flow method in a program component system, comprising a running time system (14) and several components (20, 20', ...) each having one program portion (22), said method comprising the following steps during the execution of the program portion (22) of a first component (20):
a) data acquisition by means of the running time system (14), of data of a second component (20') into said first component (20) independent of program-defined interfaces in said second component (20'); and
b) data disposal by means of the running time system (14), of data of said first component (20) into said second component (20') independent of program-defined interfaces in said second component (20').

2. The method according to claim 1, characterized in that the data transmitted during the data acquisition are transferred from a memory image portion (28) of said second component (20') into a transfer data region (34) of said first component (20), and/or that the data transmitted during the data disposal are transferred from a transfer data region (34) of said first component (20) into a memory image portion (28) of said second component (20').

3. The method according to ~~claims 1 or 2~~ Claim 1, characterized in that said data acquisition and/or data disposal is carried out without the cooperation of said second component (20').

4. The method according to ~~any one of claims 1 to 3~~ Claim 1, characterized in that said second component (20') is inactive during said data acquisition and/or said data disposal.

5. The method according to ~~any one of claims 1 to 4~~ Claim 1, characterized in that said transfer data region (34) of said second component (20') is located in a saving region during said data acquisition and said data disposal.

6. The method according to ~~any one of claims 1 to 5~~ Claim 1, characterized in that local and/or non-persistent data of said second component (20') are transmitted during said data acquisition and/or said data disposal.

7. The method according to ~~any one of claims 1 to 6~~ Claim 1, characterized in that during the execution of the program portion (22) of said first component (20) a waiting list is made indicating which of the data of said first component (20) require data disposal.

8. The method according to ~~any one of claims 1 to 7~~ Claim 1, characterized in that a called component (20, 20', ...) can directly access an access data region (30) comprising the data fields defined and/or available in said calling component (20, 20', ...).

9. The method according to ~~any one of claims 1 to 8~~ Claim 1, characterized in that the call of a component (20, 20', ...) is triggered by call information comprised in a docking point of the calling component.

10. A method of expanding a program component system comprising several components (20, 20', ...), by one further component, said method comprising the steps of:

a) searching for docking points for said further component in said program component system, which docking points correspond to an inheritance parameter determined by a definition of said further component; and

b) modifying the components (20, 20', ...) of the program component system where at least one docking point was found, by entering call information about the further component at each docking point found.

11. The method according to claim 10, characterized in that all interaction interfaces of the previous components (20, 20', ...) are predefined as potential docking points.

12. The method according to claim 10, characterized in that all interaction screen fields referenced by the previous components (20, 20', ...) and/or all print mask output fields and/or all access operations on persistent data are predefined as potential docking points.

13. The method according to ~~any one of claims 10 to 12~~ claim 10. characterized in that by entering said call information into a docking point a call of the further component from the component into which said call information was entered, is prepared.

14. The method according to ~~any one of claims 10 to 13~~ claim 10, characterized by an additional step of

c) generating at least one binary object from the definition of the further component.

15. The method according to claim 14, characterized in that a maximum of one binary object is generated for each docking point that has been found.

16. The method according to claim 15, characterized in that while generating each binary object, the memory allocation is considered in the one component (20, 20', ...) of the program component system which includes the underlying docking point.

09582771.080200